Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Informatyki, Elektroniki i Telekomunikacji
Instytut Informatyki

Języki formalne i kompilatory

Projekt zaliczeniowy

Translator języka LaTeX do HTML

Bartosz Kordek Grzegorz Zacharski

Informatyka

Studia niestacjonarne

Kraków, 2021

Spis treści

Rozdział 1

Wstęp

Celem projektu było zaimplementowanie translatora języka LaTeX do HTML. W tym celu skorzystaliśmy z gotowej biblioteki PLY [3] do skanowania i parsowania napisanej w języku Python. Kod odpowiadający za wykonanie translacji podzieliliśmy na dwie części: lekser (skanowanie) i parser (parsowanie).

Do wykonania projektu skorzystaliśmy również z oficjalnej dokumentacji języka LaTeX [2] oraz języka HTML [1].

1.1 Pliki tworzące projekt

Translator został zaimplementowany w języku Python 3. Projekt składa się z następujących plików:

- main.py - plik wejściowy programu, odpowiada za uruchomienie interpretera, wyświetlanie pomocniczych informacji w trybie debugowania oraz za uruchomienie odpowiedniego trybu działania programu
- theLexer.py - lexer, który ma za zadanie zwrócić tokeny po przeanalizowaniu kodu dokumentu latex
- theParser.py - parser, który ma za zadanie stworzyć abstrakcyjne drzewo syntaktyczne oraz zwrócic kod html
- example.tex - przykładowy plik testowy, który zostanie przetłumaczony do języka html

1.2 Uruchamianie

W celu uruchomienia tranlatora trzeba mieć zainstalowany interpreter języka Python 3 oraz w folderze projektu wpisać w terminalu komendę pokazaną na listingu 1.1:

```
python3 main.py -i example -o example
```

Listing 1.1: Uruchomienie

Parametr *i* przyjmuje nazwę pliku do translacji, natomiast paramentr *o* przyjmuje nazwę pliki html, który zostanie stworzony podczas działania programu.

Rozdział 2

Specyfikacja gramatyki

2.1 Tekst

Nie wyszczególniono specjalnych wymagań, które musi spełniać zwykły tekst w języku LaTeX , aby został przetłumaczony na język HTML.

2.2 Formatowanie tekstu

Translator obsługuje następujące znaczniki formatu LaTeX odpowiadające na formatowanie tekstu:

- *bold* - pogrubienie
- *italic* - kursywa
- *underline* - podkreślenie
- *centerline* - wyśrodkowanie
- *tabulator* - wcięcie
- *newline* - nowa linia
- *title* - tytuł

2.3 Tabela

W celu przetłumaczenia tabeli z języka LaTeX na język HTML tabela powinna spełniać powniższe założenia:

- musi być zadeklarowana poleceniem $\backslash begin\{tabular\}$ oraz zakończona $\backslash end\{tabular\}$

- po deklaracji *begin*{*tabular*} musi zostać od razu zadeklarowane obramowanie wg standardu LaTeX , np. *begin*{*tabular*}{*c c c*}
- tabele z obramowaniem muszą zostać zadeklarowane wg *begin*{*tabular*}{| *c* | *c* | *c* |} - ilość kolumn dowolna
- tabele bez obramowania muszą zostać zadeklarowane wg *begin*{*tabular*}{*c c c*} - ilość kolumn dowolna

2.4 Wyliczenie

Program obsługuje wyliczenie uporządkowane i nieuporządkowane:

- *backslashbegin*{*enumerate*} - uporządkowane
- *backslashbegin*{*itemize*} - nieuporządkowane

2.5 Grafika

Grafikę w tekście możemy zadeklarować wyłącznie używając polecenia *includegraphics* podając ściezkę do zdjęcia np: *includegraphics*{*corgi.jpg*}.

2.6 Hiperłącze

Hiperłącze w deklarujemy wyłącznie poprzez użycie *url* np. *url*{*https://github.com/bartoszkordek/AGH Jezyki-formalne-i-kompilatory*}

2.7 Sekcja, podsekcja, podpodsekcja

Sekcje, podsekcje oraz podpodsekcje deklarujemy w ten sam sposób jak w języku LaTeX np. *subsection*{*Przykladowa podsekcja*}.

Rozdział 3

Opis systemu typizacji

3.1 Lexer

Na listingu 3.1 zostały przedstawione tokeny obsługiwane przez nasz translator.

```
    tokens = (
        'AUTHOR',
        'BEGIN_DOCUMENT',
        'BEGIN_OLIST',
        'BEGIN_ULIST',
        'BEGIN_TABULAR',
        'BOLD',
        'CENTERLINE',
        'CHAPTER',
        'COLUMN_DIVIDER',
        'COLUMN_PATTERN_BORDERLESS',
        'COLUMN_PATTERN_BORDERED',
        'DOCUMENTCLASS',
        'END_DOCUMENT',
        'END_OLIST',
        'END_ULIST',
        'END_TABULAR',
        'GRAPHICS_PATH',
        'HLINE',
        'INCLUDE_GRAPHICS',
        'ITALIC',
        'ITEM',
        'LBRACE',
        'NEW_LINE',
        'NULL',
        'PARAGRAPH',
        'RBRACE',
        'ROW_END',
        'SECTION',
```

```
        'SUBSECTION',
        'SUBSUBSECTION',
        'TEXT',
        'TITLE',
        'UNDERLINE',
        'URL',
        'USE_PACKAGE',
    )
```

Listing 3.1: Tokeny

Tokeny na listingu 3.1 odpowiadają następującym wyrażeniom regularnym przedstawionym na listingu 3.2:

```
    t_AUTHOR = r'\\author'
    t_BEGIN_DOCUMENT = r'\\begin\{document\}'
    t_BEGIN_OLIST = r'\\begin\{enumerate\}'
    t_BEGIN_ULIST = r'\\begin\{itemize\}'
    t_BEGIN_TABULAR = r'\\begin\{tabular\}'
    t_BOLD = r'\\textbf'
    t_CENTERLINE = r'\\centerline'
    t_CHAPTER = r'\\chapter'
    t_COLUMN_DIVIDER = r'&'
    t_COLUMN_PATTERN_BORDERLESS = r'\{[lcr](\s[lcr])*\}'
    t_COLUMN_PATTERN_BORDERED = r'\{(\|\s[lcr]\s)+\|\}'
    t_DOCUMENTCLASS = r'\\documentclass.*'
    t_END_DOCUMENT = r'\\end\{document\}'
    t_END_OLIST = r'\\end\{enumerate\}'
    t_END_ULIST = r'\\end\{itemize\}'
    t_END_TABULAR = r'\\end\{tabular\}'
    t_GRAPHICS_PATH = r'\\graphicspath'
    t_INCLUDE_GRAPHICS = r'\\includegraphics'
    t_ITALIC = r'\\textit'
    t_ITEM = r'\\item'
    t_LBRACE = r'\{'
    t_NEW_LINE = r'\\newline'
    t_NULL = r'\0'
    t_PARAGRAPH = r'\\paragraph'
    t_RBRACE = r'\}'
    t_ROW_END = r'\\\\'
    t_SECTION = r'\\section'
    t_SUBSECTION = r'\\subsection'
    t_SUBSUBSECTION = r'\\subsubsection'
    t_TEXT = r'[\w\d\.,!?@#/\'\"<>\(\)\-+=\/^\*:;|\[\]]+'
    t_TITLE = r'\\title'
    t_UNDERLINE = r'\\underline'
    t_URL = r'\\url'
```

```
34      t_USE_PACKAGE = r'\\usepackage.*'
```

Listing 3.2: Wyrażenia regularne

Definicje tokenów nowej linii, komentarza, błędów oraz tokenów ignorowanych zostały przedstawione na listingu 3.3.

```
1       def t_newline(self, t):
2           r'\n+'
3           t.lexer.lineno += t.value.count("\n")
4
5       def t_comment(self, t):
6           r'%.*\n'
7           pass
8
9       def t_hline(self, t):
10          r'\\hline'
11          pass
12
13      t_ignore = ' '
14
15      def t_error(self, t):
16          print("Illegal character '%s'" % t.value[0])
17          t.lexer.skip(1)
```

Listing 3.3: Pozostałe tokeny

Należy zauważyć, że cała zawartość między znacznikiem "%"w formacie LaTeX, odpowiadającemu początkowi komentarza, do nowej linii, zostaje pominięta.

3.2 Parser

Każdy z dokumentów HTML'a musi zawierać znaczniki typowe dla tego formatu, których nie znajdziemy w dokumentach LaTeX. Są to: ***html***, ***head*** oraz ***body***. Jako odpowienik sekcji ***head*** przyjęliśmy sekcję preambuly w której znajdują się między innymi takie polecenia jak *usepackage*. Natomiast jak odpowiednik sekcji ***body*** odpowiada fragment kodu LaTeX objęty poleceniami *begin*{*document*} oraz *end*{*document*}.

```
1       def p_statement_preamble(self, p):
2           'statement : DOCUMENTCLASS head body'
3           p[0] = '<!DOCTYPE html>'+'\n<html lang="en">' + p[2]
4               + p[3]+'\n</html>'
5
6       def p_header(self, p):
7           '''head : USE_PACKAGE head
8                   | USE_PACKAGE'''
9           p[0] = '\n<head>' + '<meta charset="UTF-8"/>' + '\n</head>'
10
```

```
    def p_body(self, p):
        'body : BEGIN_DOCUMENT expression END_DOCUMENT'
        p[0] = '\n<body>\n' + p[2] + '\n</body>'
```

Listing 3.4: Preambuła

3.2.1 Obsługa tekstu

```
    def p_expression_text(self, p):
        '''expression : TEXT expression
                      | TEXT'''
        if len(p) == 3:
            p[0] = p[1] + " " + p[2]
        else:
            p[0] = p[1]
```

Listing 3.5: Tekst

3.2.2 Formatowanie tekstu

Nasz translator umożliwia pogrubienie, kursywę, podkreślenie, wyśrodkowanie tekstu oraz utworzenie paragrafu. Możliwe jest mieszanie stylów formatowania, np. pogrubienie z podkreśleniem. Według nowych zaleceń, do pogrubienia tekstu w HTML'u powinno się stosować znacznik **, a do kursywy **.

```
    def p_expression_bold(self, p):
        '''expression : BOLD LBRACE expression RBRACE expression
                      | BOLD LBRACE expression RBRACE'''
        if len(p) == 6:
            p[0] = '<strong>' + p[3] + '</strong>' + p[5]
        else:
            p[0] = '<strong>' + p[3] + '</strong>'

    def p_expression_italic(self, p):
        '''expression : ITALIC LBRACE expression RBRACE expression
                      | ITALIC LBRACE expression RBRACE'''
        if len(p) == 6:
            p[0] = '<em>' + p[3] + '</em>' + p[5]
        else:
            p[0] = '<em>' + p[3] + '</em>'

    def p_expression_underline(self, p):
        '''expression : UNDERLINE LBRACE expression RBRACE expression
                      | UNDERLINE LBRACE expression RBRACE'''
        if len(p) == 6:
            p[0] = '<u>' + p[3] + '</u>' + p[5]
```

```
        else:
            p[0] = '<u>' + p[3] + '</u>'

    def p_expression_centerline(self, p):
        '''expression : CENTERLINE LBRACE expression RBRACE expression
                      | CENTERLINE LBRACE expression RBRACE'''
        if len(p) == 6:
            p[0] = '<center>' + p[3] + '</center>' + p[5]
        else:
            p[0] = '<center>' + p[3] + '</center>'

    def p_expression_paragraph(self, p):
        '''expression : PARAGRAPH LBRACE expression RBRACE expression
                      | PARAGRAPH LBRACE expression RBRACE'''
        if len(p) == 6:
            p[0] = '<p>' + p[3] + '</p>' + p[5]
        else:
            p[0] = '<p>' + p[3] + '</p>'
```

Listing 3.6: Formatowanie - pogrubienie, kursywa, podkreślenie, wyśrodkowanie tekstu

Kolejną funkcjonalnością jest możliwość obsługi rozdziałów, sekcji i podsekcji parsując znaczniki *chapter*, *section*, *subsection* oraz *subsubsection*.

```
    def p_expression_chapter(self, p):
        '''expression : CHAPTER LBRACE expression RBRACE expression
                      | CHAPTER LBRACE expression RBRACE'''
        if len(p) == 6:
            p[0] = '<h1>' + p[3] + '</h1>' + p[5]
        else:
            p[0] = '<h1>' + p[3] + '</h1>'

    def p_expression_section(self, p):
        '''expression : SECTION LBRACE expression RBRACE expression
                      | SECTION LBRACE expression RBRACE'''
        if len(p) == 6:
            p[0] = '<h2>' + p[3] + '</h2>' + p[5]
        else:
            p[0] = '<h2>' + p[3] + '</h2>'

    def p_expression_subsection(self, p):
        '''expression : SUBSECTION LBRACE expression RBRACE expression
                      | SUBSECTION LBRACE expression RBRACE'''
        if len(p) == 6:
            p[0] = '<h3>' + p[3] + '</h3>' + p[5]
        else:
            p[0] = '<h3>' + p[3] + '</h3>'

```

```
    def p_expression_subsubsection(self, p):
        '''expression : SUBSUBSECTION LBRACE expression RBRACE expression
                      | SUBSUBSECTION LBRACE expression RBRACE'''
        if len(p) == 6:
            p[0] = '<h4>' + p[3] + '</h4>' + p[5]
        else:
            p[0] = '<h4>' + p[3] + '</h4>'
```

Listing 3.7: Rozdziały, sekcje i podsekcje

Przejście do nowej linii (hard break) jest obsłużone poleceniem *newline*.

```
    def p_expression_newline(self, p):
        '''expression : NEW_LINE expression
                      | NEW_LINE'''
        if len(p) == 3:
            p[0] = '<br/>' + p[2]
        else:
            p[0] = p[1]
```

Listing 3.8: Nowa linia

Translator parsuje również znacznik *title* odpowiadający za utworzenie tytułu.

```
    def p_title(self, p):
        'expression : TITLE LBRACE TEXT RBRACE'
        p[0] = '<i>' + p[3] + '</i>'
```

Listing 3.9: Tytuł

3.2.3 Tabela

Z obramowaniem

```
    def p_expression_table_bordered(self, p):
        '''expression : BEGIN_TABULAR COLUMN_PATTERN_BORDERED
    tablerowbordered END_TABULAR expression | BEGIN_TABULAR
    COLUMN_PATTERN_BORDERED tablerowbordered END_TABULAR'''
        if len(p) == 6:
            p[0] = '<table style="border: 1px solid black; border-collapse:
    collapse;">' + p[3] + '</table>' + p[5]
        else:
            p[0] = '<table style="border: 1px solid black; border-collapse:
    collapse;">' + p[3] + '</table>'

    def p_tablerowbordered(self, p):
        '''tablerowbordered : tablecolumnbordered ROW_END tablerowbordered |
     tablecolumnbordered'''
        if len(p) == 4:
            p[0] = '<tr>' + p[1] + '</tr>' + p[3]
```

```
12          else:
13              p[0] = '<tr>' + p[1] + '</tr>'
14
15      def p_tablecolumnbordered(self, p):
16          '''tablecolumnbordered : expression COLUMN_DIVIDER
    tablecolumnbordered | expression'''
17          if len(p) == 4:
18              p[0] = '<td style="border: 1px solid black; border-collapse:
    collapse;">' + p[1] + '</td>' + p[3]
19          else:
20              p[0] = '<td style="border: 1px solid black; border-collapse:
    collapse;">' + p[1] + '</td>'
```

Listing 3.10: Tabela z obramowaniem

Bez obramowania

```
1       def p_expression_table_borderless(self, p):
2           '''expression : BEGIN_TABULAR COLUMN_PATTERN_BORDERLESS
    tablerowborderless END_TABULAR expression | BEGIN_TABULAR
    COLUMN_PATTERN_BORDERLESS tablerowborderless END_TABULAR'''
3           if len(p) == 6:
4               p[0] = '<table>' + p[3] + '</table>' + p[5]
5           else:
6               p[0] = '<table>' + p[3] + '</table>'
7
8       def p_tablerowborderless(self, p):
9           '''tablerowborderless : tablecolumnborderless ROW_END
    tablerowborderless | tablecolumnborderless'''
10          if len(p) == 4:
11              p[0] = '<tr>' + p[1] + '</tr>' + p[3]
12          else:
13              p[0] = '<tr>' + p[1] + '</tr>'
14
15      def p_tablecolumnborderless(self, p):
16          '''tablecolumnborderless : expression COLUMN_DIVIDER
    tablecolumnborderless | expression'''
17          if len(p) == 4:
18              p[0] = '<td>' + p[1] + '</td>' + p[3]
19          else:
20              p[0] = '<td>' + p[1] + '</td>'
```

Listing 3.11: Tabela bez obramowania

3.2.4 Wyliczenie

Konstrukcja parsera wyliczeń umożliwia wykonywanie zagnieżdżeń.

Uporządkowane

```
    def p_expression_ordered_list(self, p):
        '''expression : BEGIN_OLIST listitems END_OLIST expression
                      | BEGIN_OLIST listitems END_OLIST'''
        if len(p) == 5:
            p[0] = '\n<ol>' + p[2] + '\n</ol>' + p[4]
        else:
            p[0] = '\n<ol>' + p[2] + '\n</ol>'
```

Listing 3.12: Wyliczenie uporządkowane

Nieuporządkowane

```
    def p_expression_unordered_list(self, p):
        '''expression : BEGIN_ULIST listitems END_ULIST expression
                      | BEGIN_ULIST listitems END_ULIST'''
        if len(p) == 5:
            p[0] = '\n<ul>' + p[2] + '\n</ul>' + p[4]
        else:
            p[0] = '\n<ul>' + p[2] + '\n</ul>'
```

Listing 3.13: Wyliczenie nieuporządkowane

3.2.5 Grafika

Umieszczenie grafiki jest możliwe dzięki znacznikowi *includegraphics* w LaTeX, który jest parsowany na znacznik <*img*> w HTML, gdzie atrybut *src* stanowi ścieżka do pliku umieszczona w nawiasach wąsatych w dokumencie LaTeX.

```
    def p_expression_includegraphics(self, p):
        '''expression : INCLUDE_GRAPHICS TEXT LBRACE expression RBRACE
    expression
                      | INCLUDE_GRAPHICS LBRACE expression RBRACE expression
                      | INCLUDE_GRAPHICS LBRACE expression RBRACE'''
        if len(p) == 7:
            attributes = p[2][1:-1]
            p[0] = '''<img src="''' + p[4] + '''"''' + attributes + '''>'''
    + p[6]
        elif len(p) == 6:
            p[0] = '''<img src="''' + p[3] + '''">''' + p[5]
        else:
            p[0] = '''<img src="''' + p[3] + '''">'''
```

Listing 3.14: Frafika

3.2.6 Hiperłącze

Zamieszczenie hiperłącza w formacie LaTeXjest możliwe dzięki znacznikowi *url*, zawierającego w nawiasach wąsatych adres do strony. Parsowanny jest on na HTML'owy znacznik <*a*> z atrybutem *href* zawierającego adres.

```
    def p_expression_url(self, p):
        '''expression : URL LBRACE expression RBRACE expression
                      | URL LBRACE expression RBRACE'''
        if len(p) == 6:
            p[0] = '<a href=' + p[3] + '>' + p[3] + '</a>' + p[5]
        else:
            p[0] = '<a href=' + p[3] + '>' + p[3] + '</a>'
```

Listing 3.15: Hiperłącze

Rozdział 4

Uzasadnienie wyboru generatora

Podstawowym wymaganiem w projekcie była konieczność zastowania generatora parserów i skanerów. Z zaproponowanych na zajęciach bibliotek (PLY, SLY oraz ANTLR) zdecydowaliśmy się na wybór biblioteki PLY [3], ze względu na jej popularność stosowania, dostępność i wysoką jakość dokumentacji. Ponadto chcieliśmy również zdobyć kolejne doświadczenie w implementacji kodu z wykorzystaniem języka Python. Na niekorzyść wykorzystania biblioteki SLY przemawiała niska jakość dokumentacji oraz brak przykładów.

Rozdział 5

Napotkane problemy

Podczas realizacji projektu napotkaliśmy następujące problemy do rozwiązania:

- przejście do nowej linii
- translacja tabeli

5.1 Przejście do nowej linii

W języku LateX przejście do nowej linii jest możliwe dzięki następujący sposób: wykorzystując znacznik newline, podwójnemu backslashowi, oraz wykonaniem podwójnego prześcia do nowej linii za pomocą klawisza Enter. W rezultacie chcieliśmy otrzymać znacznik "br"w HTMLu. Nie napotkaliśmy większych problemów z translacją znacznika ńewline". Podwójny backslash również działał poprawnie dopóki nie zaimplementowaliśmy translacji tabeli, w której ten sam znacznik oznacza jej koniec. Ze względu na ten problem musieliśmy z niego zrezygnować. Problem nastąpił również z przejściem do nowej linii przy pomocy dwukrotnego wciśniecia klawisza Enter. W tym przypadku każde przejście do nowej linii było traktowane jako przejście pojedyncze, a podwójne było pomijane. W rezultacie oznaczało to, że nie mogliśmy otrzymać znacznika "br"w HTMLu. Po usunięciu definicji tokena ńewline", znacznik "br"był produkowany, lecz program generował ostrzeżenie (brak znajomości tokena) w przypadku wystąpienia pojedynczego przejścia. Plik w formacie HTML był jednak poprawnie generowany. Ze względu na ostrzeżenie, pomysł został porzucony.

5.2 Translacja tabeli

Podstawowym problemem był opisany powyżej znacznik przejścia do nowej linii, który w przypadku tabeli oznacza jej koniec. Ponadto tabela jest na tyle złożoną sktrukturą, że nie jesteśmy w stanie zapewnić obsługi każdego z jej przypadków. Uprościliśmy więc translację do tabeli w pełni obramowanej lub nieobramowanej.

Listings

Bibliografia

[1] HTML 5.2. *W3C Recommendation*. 2021. URL: https://www.w3.org/TR/html52/.

[2] Overleaf. *Official Documentation*. 2021. URL: https://www.overleaf.com/learn.

[3] PLY. *Official Documentation*. 2021. URL: https://ply.readthedocs.io/en/latest/.